Converting logs into metrics

As an experienced Splunk admin, you have followed data onboarding best practices and have a well-formatted JSON event that you hand off to a data consumer.

```
{
  "device": {
    "deviceId": "127334527887",
    "deviceSourceId": "be:f3:af:c2:01:f1",
    "deviceType": "IPGateway"
  },
  "timestamp": 1489095004000,
  "rawAttributes": {
    "WIFI_TX_2_split": "325,650,390,150,150,780,293,135,325",
    "WIFI_RX_2_split": "123,456,123,456,123,456,123,456,123",
    "WIFI_SNR_2_split": "32, 18, 13, 43, 32, 50, 23, 12, 54",
  }
}
```

The consumer comes back to you, however, with the following complaint:

"These events are from our router. The device field at the top describes the router itself, and then the rawAttributes describes all of the downstream devices that connect to the router and their respective performance values like transmit, receive, and signal to noise values. We want to be able to report on these individual downstream devices over time and associate those individual devices with the router that serviced them as well as investigate the metrics over time. We use this data to triage customer complaints and over time, improve the resiliency of our network."

This means that multi-device single events must be transformed into distinct values. Worse, every single search that works with this data will need to contain all that same transformation logic, every time. You could do some ingestion-time props and transforms work to pre-format the data, coupled with some summary stats searches to transform these events for consumption. But this solution required more valuable CPU cycles used, slower search performance overall, additional resource contention, and really just a bad day for the data consumer. You need a better solution.

---

Build metrics with dimensions

You need to pre-process the data. Building metrics with dimensions for each of the consumer devices allows them to rely on super-fast mstats for their near real-time reporting. With familiar search processing language, you can apply the needed transformations in the stream, before the data is indexed. Doing so removes complexity from the data, reduces search and index time resource consumption, improves data quality, and in the case of this customer, reduces their mean time to identify problems in their environment because they're not waiting for actionable data to be generated.
This example uses data sent from a heavy forwarder to Splunk DSP firehose. You may need to change the data source and applicable Splunk Technical Add-ons to fit your environment.

1. From the Stream Processor Service home page, click Get Started.
2. Click Sources.
4. Click the node icon on the right of the data source box to add a function.

5. Using the function tray on the right, add the following functions sequentially and pass the following arguments:
   - Function 1: WHERE source="carrierjson"
   - Function 2: EVAL bodyjson = from_json_object(tostring(body))
   - Function 3: EVAL attr=ucast(bodyjson.rawAttributes, "map<string,string>", null),
     tx=split(attr.WIFI_TX_2_split, ","),
     rx=split(attr.WIFI_RX_2_split, ","),
     snr=split(attr.WIFI_SNR_2_split, ","),
     macs=split(attr.ClientMac_split, ","
   - Function 4: EVAL deviceRange = mvrange(0,length(macs)-1)
   - Function 5: EVAL temp=for_each(iterator(deviceRange, "i"), {"timestamp": timestamp,  
     "host": spath(bodyjson, "device.deviceId"),  
     "attrs":{"default_dimensions": {"mac": mvindex(macs, i)},  
     "body": [{"name": "wifi_tx", "value": mvindex(tx, i)},  
     {"name": "wifi_rx", "value": mvindex(rx, i)},  
     {"name": "wifi_snr", "value": mvindex(snr, i)}]})

   In this eval step, click +Add to add the body parameters, one at a time, before clicking the node icon again to add to the mvexpand function, which is next.

   - Function 6: MVEXPAND limit=0 temp
   - Function 7: EVAL host=tostring(temp.host), attributes=ucast(temp.attrs, "map<string,any>", null),
     body=temp.body, kind="metric"
   - Function 8: SELECT host, timestamp, attributes, body, kind, source
   - Function 9: INTO splunk_enterprise_indexes("Splunk_Connection", "carrier_metrics", "main");

6. Verify with a search that the metrics are flowing in.
7. Using the analytics workbench, build charts from these metrics to verify that the MAC address and host...
dimensions are both available for splitting.

Results

In ten commands, you have gone from a condensed multivalue plain text event into an expanded metric-based statistic, ready to use within seconds of arriving at Splunk Stream Processor Services. At first glance, this may seem like a lot of work. But when you consider that had you not done this, the user would have had to figure out the best way to split up all those events and that logic would have to be used every single time that data is required. This is a significant barrier removed and it'll make you a hero to your users.

Next steps

These additional Splunk resources might help you understand and implement these recommendations:

- Community Case Study: Complex data transformation using real time-time stream processing
- Blog: What is Stream Processing?
- Docs: Stream Processor Service
- Docs: Data Stream Processor